

Visual Closed-Loop Control for Pouring Liquids

Connor Schenck, Dieter Fox
University of Washington

Abstract—Pouring a specific amount of liquid is a challenging task. In this paper we develop methods for robots to use visual feedback to perform closed-loop control for pouring liquids. We propose both a model-based and a model-free method utilizing deep learning for estimating the volume of liquid in a container. Our results show that the model-free method is better able to estimate the volume. We combine this with a simple PID controller to pour specific amounts of liquid, and show that the robot is able to achieve an average 38ml deviation from the target amount. To our knowledge, this is the first use of raw visual feedback to pour liquids in robotics.

I. INTRODUCTION

The last years have seen dramatic improvements in robotic capabilities relevant to household tasks such as putting items into a dishwasher [1], folding and ironing clothing [2], [3], and cleaning surfaces [4]. So far, however, robots have not been able to robustly perform household tasks involving liquids, such as pouring a glass of water. Solving such tasks requires both robust control and detection of liquid during the pouring operation. Humans often are not very accurate at this, requiring specialized containers to measure a specific amount of liquid. Instead people often use vague, relative terms such as “Pour me a half cup of coffee” or “Just a little, please.” While there has been recent success in robotics on controlling a manipulator to pour liquids simulated by small balls [5] and on detecting liquids using optical flow or deep learning [6], [7], the task of pouring certain amounts of actual liquids has not been addressed.

In this paper, we introduce a framework that enables robots to robustly pour specific amounts of a liquid into containers typically found in a home environment, such as coffee mugs, cups, glasses, or bowls. We achieve this in the most general setting, without requiring specialized hardware, such as highly accurate force sensors for measuring the amount of liquid held by a robot manipulator, scales placed under the target container, or sensors designed for detecting liquids. However, while we avoid requiring specialized environmental augmentation, our investigation is on how accurate a robot could pour under relatively controlled conditions, such as having been able to train on the target containers.

The intuition behind our approach is based on the insight that people strongly rely on visual cues when pouring liquids. For example, a health study revealed that the amount of wine people pour into a glass is strongly biased by visual factors such as the shape of the glass or the color of the wine [8]. We thus propose a framework that uses visual feedback in a closed-loop pouring controller. Specifically, we train a deep neural network structure to estimate the amount of liquid in a cup from raw visual data. Our network structure has two



Fig. 1: The Baxter robot used in our experiments. In its right gripper it holds the cup used as the source container. On the table in front of the robot are the three target containers (from left to right): the small bowl, tan mug, and redgray mug.

stages. In the first stage, a network detects which pixels in a camera image contain water. The output of the detection network is fed into another network that estimates the amount of liquid already in the container. This amount is used as real-time feedback in a PID controller that is tasked to pour a desired amount of water into a cup.

To generate labeled data needed for the neural networks, we developed an experimental setup that uses a thermal camera calibrated with an RGBD camera to automatically label which pixels in the color frames contain (heated) water. Experiments with a Baxter robot pouring water into three different containers (two mugs and one bowl) indicate that this approach allows us to train deep networks that provide sufficiently accurate volume estimates for the pouring task.

Our main contributions in this paper are (1) an overall framework for determining the amount of liquid in a container for real-time control during a pouring action; (2) the use of thermal imagery to generate ground truth data for pixel level labeling of (heated) water; (3) a deep neural network that uses such labels to detect liquid pixels in raw color images; (4) a model-based method to determine the volume of liquid in a target container given pixel-wise liquid detection; (5) a neural network to regress to the volume of liquid given pixel-wise liquid detections as input; and (6) an extensive evaluation that shows that our methodology is suitable for control by deploying it on a robot for use in a pouring task.

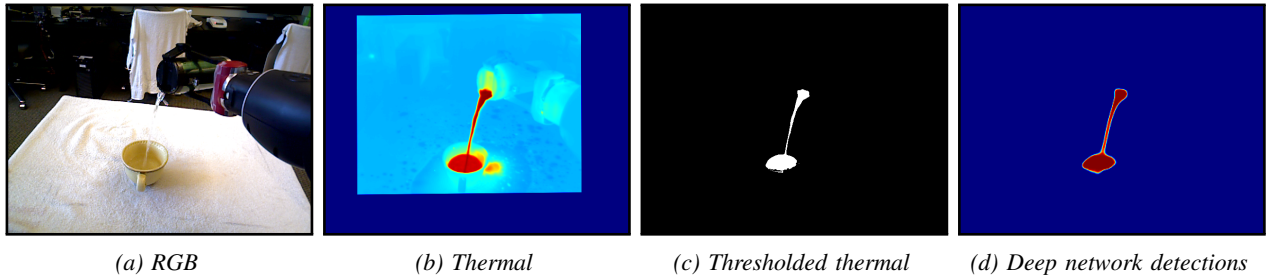


Fig. 2: An example frame from a pouring sequence. The left image shows the color camera from the robot’s perspective, the left-middle image shows a heatmap of the thermal camera after it has been registered to the color camera, the right-middle image shows the water labeled by thresholding the thermal image, and the right image shows the output of the detection network based on the RGB input.

II. RELATED WORK

There is prior work related to robotic pouring, however, most of it either uses coarse simulations disconnected from real liquid perception and dynamics [9], [5] or constrained task spaces that bypass the need to perceive and reason directly about liquids [10], [11], [12], [13], [14]. Additionally, all of these works with the exception of [14] pour the entire contents of the source container into the target container, with the focus on other factors such as spillage or the overall motion trajectory. In contrast, in this work we focus primarily on pouring a specific amount of liquid from the source into the target rather than simply emptying the source container into the target. To do this, the robot requires some method for estimating the volume of liquid in the target. Roza *et al.*[14] utilized force sensors in the robot’s arm to measure how much had been poured out, however this requires a robot with very precise torque sensors, which are not available on our Baxter robot. In our own prior work [15] we placed a digital scale under the target container. But this method presents many of its own challenges, such as delay in the scale measurement (often 1-2 seconds) and no information about where the liquid is or how it is moving. Humans, on the other hand, are able to accomplish this task purely from visual feedback, which strongly suggests that robots should be able to as well.

There is some prior work related to directly perceiving liquids from sensory feedback [16], [17], [18]. Work by Yamaguchi and Atkeson [6] utilizes optical flow to detect liquids as they flow from a source into a target container. However, for the tasks in this paper, the robot must also be able to detect standing water with no motion, for which optical flow is poorly suited. Instead, we build on our own prior work relating to liquid detection in simulation [7]. We developed a method utilizing fully-convolutional neural networks [19] to label pixels in an image as either *liquid* or *not-liquid*. Here we utilize the recurrent network with long short-term memory (LSTM) layers [20] that we used in that work to detect and label liquid in an image.

III. TECHNICAL APPROACH

A. Task Overview

In this paper, the robot is tasked with pouring a specific amount of liquid from a source container into a target container. This task is more difficult than prior work on robotic pouring which primarily focuses on pouring all the

contents of the source container into the target container, whereas we focus on pouring only a limited amount from the source. To accomplish this, the robot must use visual feedback to continuously estimate the current volume of liquid in the target container. Our approach has 3 main components: First the robot detects which pixels in its visual field are liquid and which are not. Next the robot uses these detections to estimate the volume of liquid in the target container. Finally, the robot feeds these volume estimates into a controller to pour the liquid into the target. Figure 3 shows a diagram of this process. We structure the problem in this manner as opposed to simply training one end-to-end network as it allows us to train and evaluate each of the individual components of the system, which can give us better insight into its operation.

B. Pixel-Wise Liquid Detection

In order for both the model-based and model-free volume estimation methods to work, the robot must classify each pixel in the image as *liquid* or *not-liquid*. We developed two methods for acquiring these pixel labels: a thermographic camera in conjunction with heated water, and a fully-convolutional neural network[19] with color images. While the thermal camera works well for generating pixel labels, it is also rather expensive and must be registered to an RGBD sensor. In our prior work [7], we developed a method for generating pixel labels on simulated data for liquid from color images only, which we briefly describe here.

Given color images, we train a convolutional neural network (CNN) to label each pixel as *liquid* or *not-liquid* (we use the thermal camera to acquire the ground truth labeling). The network is fully-convolutional, that is, all the learned layers are convolution layers with no fully-connected layers. The output of the network is a heatmap over the image, with real values in the range $[0, 1]$, where higher values indicate a higher likelihood of liquid. In [7] we tested 3 network structures and found that a recurrent network utilizing a long short-term memory (LSTM) layer [20] performs the best of the 3. Here we use the LSTM-CNN from that paper, which is shown in the top row of Figure 3. We refer the reader to [7] for more details.

C. Volume Estimation During Pouring Sequences

We propose two different methods for estimating the volume of liquid in a target container. The first is a model-based method, which assumes we have access to a 3D model

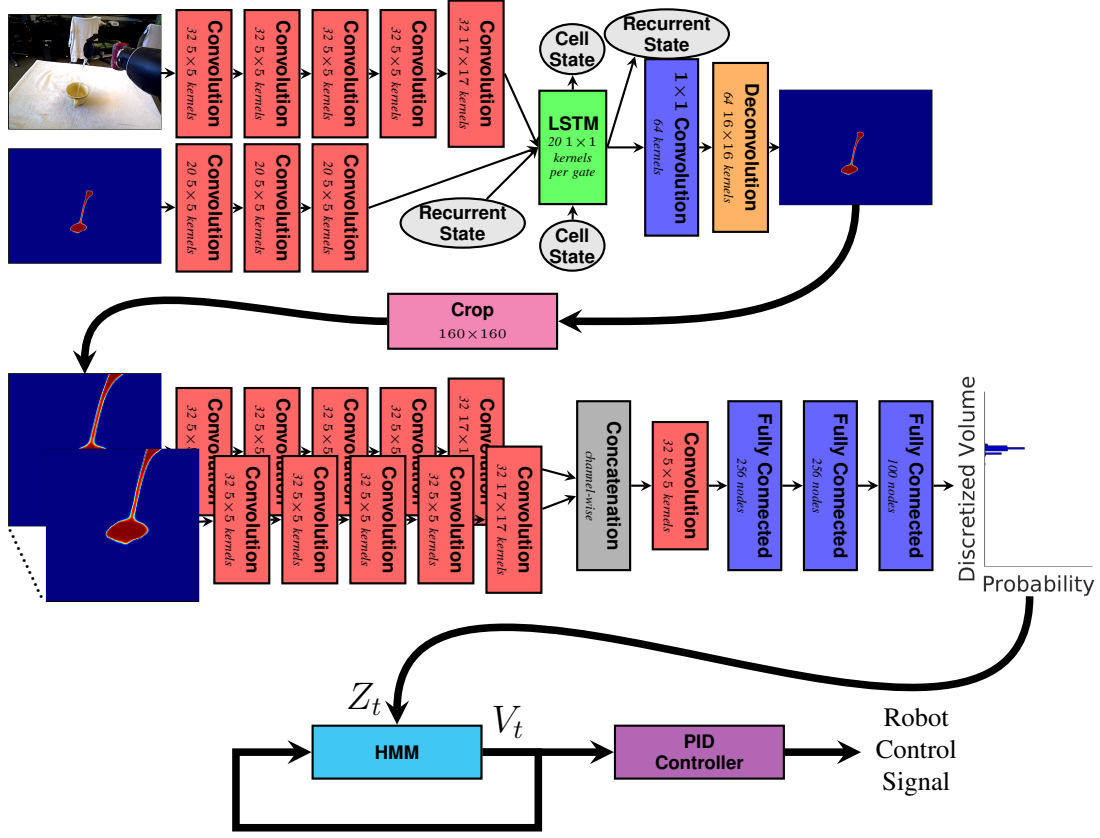


Fig. 3: The entire robot control system using the recurrent neural network for detections and the multi-frame network for volume estimation. The recurrent detection network (top) takes both the color image and its own detections from the previous time step and produces a liquid detection heatmap. The multi-frame network (center) takes a sequence of detections cropped around the target container and outputs a distribution over volumes in the container. The output of this network is fed into a HMM, which estimates the volume of the container. This is passed into a PID controller, which computes the robot’s control signal.

of the target container and infers the height of the liquid based on the camera pose and binary pixel labels. The second is a model-free method that trains a neural network to regress to the volume of liquid in the target container given labeled pixels.

1) *Model-Based Volume Estimation:* Our model-based method for estimating the volume of liquid in a target container assumes we have a 3D model of the container and that we can use the pointcloud from our RGBD sensor to find its pose in the scene. To determine the volume of liquid in the container, we first acquire the pixel-wise liquid labels as described in the previous section. Next we use these classifications to compute the height l_t of the liquid in the container at time t . If we assume that the liquid is resting level in the container¹, then there is a one-to-one correspondence between the height of the liquid and the volume, and we can use the 3D model to compute the volume given the height. We use a discrete Bayes’s filter with observations only to estimate a probability distribution L_t over l_t for all timesteps t . That is, for all t , we wish to estimate $P(L_t|L_{1:t-1}, z_{1:t})$ where $z_{1:t}$ are all the observations (pixel labels) up to time t . We make the Markovian assumption that each state is conditionally independent of all prior observations given the

previous state, thus the resulting Bayes filter is equivalent to a corresponding hidden markov model (HMM). We can estimate the posterior distribution over l_t using Bayes’s rule as follows:

$$P(l_t|L_{t-1}, z_t) \propto P(z_t|l_t, L_{t-1})P(l_t|L_{t-1}).$$

Since we make the conditional independence assumption, we can drop the L_{t-1} term from the conditional in the observation probability, resulting in $P(z_t|l_t)$. We compute $P(l_t|L_{t-1})$ as the expectation of the previous distribution times the transition probability, i.e.,

$$P(l_t|L_{t-1}) = \sum_j P(l_t|l_{t-1} = j)P(l_{t-1} = j) \quad (1)$$

where $P(l_t|l_{t-1} = j)$ is the probability of transitioning from height l^j and $P(l_{t-1} = j)$ is the prior probability of l^j at time $t - 1$.

We can compute the observation probability as

$$P(z_t|l_t) = \sum_{i \in \mathcal{I}_t} P(p_i^i|l_t)$$

where \mathcal{I}_t is the set of all pixels that see the inside of the target container, as determined by the model’s pose in the scene. We make the naive Bayes assumption that each pixel is conditionally independent of all the others given l_t , which, while not technically correct, works well in practice. To compute the observation probability of an individual pixel

¹While not strictly true throughout the entire duration of a pour, this assumption still allows for a rough measurement.

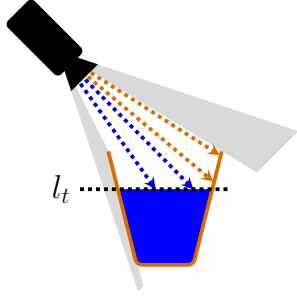


Fig. 4: Diagram of the thermal camera looking down into the target container. The blue shows where the liquid is expected to be given l_t . The blue lines show pixels expected to be classified as liquid and the orange lines show pixels that are expected to be classified as not-liquid. The gray shows pixels outside the container and not used by our algorithm.

p^i being either liquid or not liquid given the height of the surface of the water, we compare what the robot would expect to see if l_t were the true height to what the observed pixel labels are. To do this, we use the pose of the model to fit a plane to the surface height l_t and project that surface back into the camera's pixel space, generating expected pixel labels. An example of this is shown in Figure 4. Based on this, we set $P(p_t^i | w(p^i, l_t))$ as follows:

		Observed	
		Liquid	Not-liquid
Expected	Liquid	90%	10%
	Not-liquid	20%	80%

where $w(p^i, l_t)$ is the expected label at pixel p^i for l_t . We allow slightly more error for classifying a pixel as *liquid* when it is above the level of the water due to the stream of liquid falling from the source container during the pour.

We discretize the height L_t into 1000 values. After computing the distribution over $P(L_t | L_{t-1}, z_t)$, we take the median as the height of the liquid, and thus the volume, at each time step.

2) *Model-Free Volume Estimation*: Our model-free method replaces the object pose inference of the model-based method with a neural network. The neural network takes in pixel labels and produces a volume estimate. We use only the output of the detection network described in section III-B for the pixel labels, so we directly feed the heatmap over the pixels into the volume estimation network. We also evaluate adding as inputs either the color or depth images, which we append channel-wise to the pixel labels before feeding into the network. We crop the input to the network around the target container.

The output of the volume estimation network is a discrete distribution over a class label. We treat this as the observation for a discrete Bayes filter and process it in a similar manner as the previous section (notably we make the Markovian independence assumption). We can compute the distribution over volumes as

$$P(v_t | V_{t-1}, z_t) \propto P(z_t | v_t) P(v_t | V_{t-1})$$

where V_t is a distribution over the volume of liquid v_t in the target container at time t . We compute $P(v_t | V_{t-1})$ in the same manner as $P(l_t | L_{t-1})$ in the previous section.

To compute $P(z_t | v_t)$, we treat the output of the network as a probability distribution over a discrete observation. There are multiple methods we can utilize to compute the observation probability from the network output. For example, we could take the maximum probability and consider that value the observed value. However, a more robust and principled method would be to compute the expectation of the observation probability, taking the entire distribution output by the network into account. We can compute this as

$$E[P(z_t | v_t)] = \sum_i P(z_t = i | v_t) P(z_t = i)$$

where $P(z_t = i | v_t)$ is the probability that the observation is i given the volume is v_t , and $P(z_t = i)$ is the probability of state i derived from the output of the network. We use the median of $P(V_t | V_{t-1}, z_t)$ as the volume at time t .

We evaluated three different network architectures: a single-frame CNN, a multi-frame CNN, and a recurrent LSTM CNN. We use the Caffe deep learning framework [21] to implement our networks

Single-Frame CNN: The single-frame network is a standard CNN that takes as input a single image. It then passes the image through 5 convolution layers, each of which is followed by a max pooling and rectified linear layer. Every layer has a stride of 1 except for the first 3 max pooling layers. It passes the result through 3 fully connected layers, each followed by a rectified linear layer. These last 3 layers are also followed by dropout layers during training, with a drop rate of 10%. The single-frame network (CNN) is similar to the multi-frame network shown in the center row of Figure 3, with the exception that it only takes a single frame and does not have the concatenation layer or the convolution layer immediately following it.

Multi-Frame CNN: The multi-frame network (MF-CNN) is shown in the center row of Figure 3. It takes as input a set of temporally sequential images. Each image is passed independently through the first 5 layers of the network, which are identical to the first 5 convolutional layers in the single-frame network. Next, the result of each image is concatenated channel-wise and passed through another convolution layer (which is also followed by max pooling and rectified linear layers). This is then fed into 3 fully connected layers, which are identical to the last 3 layers of the single-frame CNN.

Recurrent LSTM CNN: The LSTM-CNN is identical to the single-frame network, with the exception that we replace the first fully connected layer with the LSTM layer. In addition to the output of the convolution layers, the LSTM layer also takes as input the recurrent state from the previous timestep, as well as the cell state from the previous timestep. Each gate in the LSTM layer is a 256 node fully connected layer. Please refer to Figure 1 of [22] for a detailed layout of the LSTM layer.

D. Robot Controller

For this paper, we want to investigate whether, given good real-time feedback, pouring can be performed with a simple controller. We place a table in front of the robot, and on the table we place the target container. We fix the source

container in the robot’s right gripper and pre-fill it with a specific amount of water not given to the robot. We also fix the robot’s arm such that the source container is above and slightly to the side of the target container.

To pour, the robot controls the angle of its wrist joint, thus directly controlling the angle of the source container. We use a modified PID controller to execute the pour. The robot first tilts the container to a pre-specified angle (we use 75 degrees from vertical), then begins running the PID controller, using the difference between the target volume and the current volume in the target container as its error signal. Since pouring is a non-reversible task (liquid cannot return to the source once it has left), we set the integral gain to 0, and we set the proportional and derivative gains to $\frac{0.01\pi}{180}$ and $\frac{0.2\pi}{180}$ respectively. Once the target volume has been reached, the robot rotates the source container until it is vertical once again.

E. Implementation Details

1) *Finding the Container in the Scene:* Both our model-based and model-free methods require finding the target container on the table in front of the robot (though only the model-based needs a 3D model). To find the container, we use the robot’s RGBD camera to capture a pointcloud of the scene in front of the robot and then utilize functions built-in to the PointCloud Library (PCL) [23] to find the plane of the table and cluster the points on top of the table. To acquire the pose for the model-based method, we use iterative closest points to find the 3D pose of the model in the scene. Next we use this pose to label each pixel in the image as either *inner* (inside of the container), *outer* (outside of the container), or *neither*.

2) *Classifying Pixels with the Thermographic Camera:* We use a thermal camera in combination with water heated to approximately 93°Celsius to get the ground truth pixel labels for the liquid. To register the thermal image to the color image, we use a paper checkerboard pattern attached to a 61×61 centimeter metal aluminum sheet. We then direct a small, bright spotlight at the pattern, causing a heat differential between the white and black squares, which is visible as a checkerboard pattern in the thermal image and we use OpenCV’s built-in function for finding corners of a checkerboard to find correspondence points and compute an affine transformation². We use an adaptive threshold based on the average temperature of the pixels associated with the target container (which includes the pixels for the liquid in the container). The result of this is a binary image with each pixel classified as either *liquid* or *not-liquid*. Figure 2 shows a color image, its corresponding thermal image transformed to the color pixel space, and a simple temperature threshold of the thermal image. Note that the thermal camera provides quite reliable pixel labels for liquid detection with minimal false positives

²While there has been prior work on performing full registration between thermal and color images [24], because the depth of the pour is fixed, we opted for this simpler approach. We cannot simply use the depth sensor because water does not appear on the depth image.

3) *Acquiring Ground Truth Volume Estimates for Training and Evaluation:* In order to train our networks in the previous section, and to evaluate both our model-based and model-free methods, we need a baseline ground truth volume estimation. To generate this baseline, we utilize the thermal camera in combination with the model-based method described in section III-C.1. However, since this analysis can be done *a posteriori* and does not need to be real-time, we can use the benefit of hindsight to improve our estimates, i.e., future observations can improve the current state estimate. While we acknowledge that this method does not guarantee perfect volume estimates, the combined accuracy of the thermal camera and after-the-fact processing yield robust estimates suitable for training and evaluation.

To compute this baseline we replace the forward method for HMM inference described in section III-C.1 with Viterbi decoding [25]. We replace the summation in equation 1 in the computation of the prior $P(l_t)$ with a *max* to compute the probability of each sequence. We use a corresponding *argmax* to compute the previous state from the current state, starting at the last time step and working backwards. At the last time step, we start with the most probable state. Thus using this method we can generate a reliable ground truth estimate of the volume of liquid in the target container over the duration of a pouring sequence to use for training our learning algorithms and evaluating our methodology.

IV. EXPERIMENTS & RESULTS

A. Robotic Platform

All of our experiments were performed on our Rethink Robotics Baxter Research Robot, shown in Figure 1. It is equipped with two 7-dof arms, each with an electric parallel gripper. For the experiments in this paper, we use exclusively the right arm. The robot has an Asus Xtion Pro mounted on its upper-torso, directly below its screen, which includes both an RGB color camera and a depth sensor, each of which produce 640×480 images at 30Hz. Mounted on the robot immediately above the Xtion sensor is an Infrared Cameras Inc. 8640P Thermal Imaging Camera, which reads the temperature of the image at each pixel and outputs a 640×512 image at 30Hz.

B. Experimental Setup

For all experiments, the robot poured from the cup shown in its gripper in Figure 1. We used three target containers, also shown in Figure 1. We collected a dataset of pours using this setup in order to both train and evaluate our methodologies. We collected a total of 279 pouring sequences, in which the robot attempted to pour 250ml of water into the target using the thermal camera with the model-based method, with the initial amount in the varied between 300ml, 350ml, and 400ml. Each sequence lasted exactly 25 seconds and was recorded on both the thermal and RGBD cameras at 30Hz. We randomly divided the data 75%-25% into train and evaluation sets. After the data was collected, we used the thermal images to generate ground truth pixel labels as well as we used the Viterbi decoding method described in section

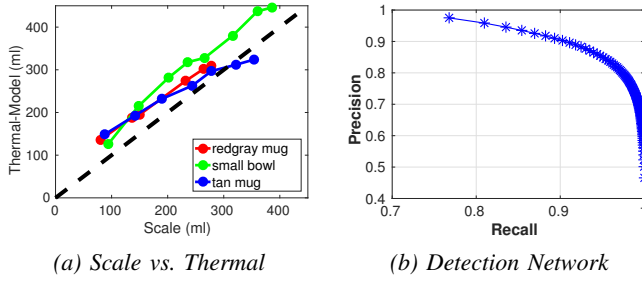


Fig. 5: The left plot shows the scale reading compared to the thermal camera with the model based method for each of the target containers. The right plot shows the pixel-wise precision and recall curves for the liquid detection network.

III-E.3 to generate ground truth volume estimates, which we compare against for the remainder of this section. We use this ground truth estimate to directly infer $P(v_t|V_{t-1})$ used in the HMMs described previously.

C. Validating Thermographic Ground Truth Methodology

Before we can evaluate our methodologies, we must first verify that our method for generating ground truth volume estimates is accurate. We can compare a static volume measurement with a scale to static estimates from the thermal camera combined with the model-based method to gauge the accuracy of our method. Figure 5a shows a comparison between measurements from a scale (x-axis) and the corresponding measurement from the thermal camera with model (y-axis) for each of the three target containers. The black dashed line shows a 1:1 correspondence for reference. From the figure it is clear that the model-based method overestimates the volume for each container. In order to make our baseline as accurate as possible, we fit a linear model for each container and use that to calibrate the baseline ground truth estimates described in section III-E.3.

D. Evaluating the Detection Network

Next we must verify that the neural network we trained to labels pixels as *liquid* or *not-liquid* from color images is accurate enough to utilize for volume estimation. Our prior work [7] showed that neural networks can label liquid pixels in an image reasonably well on data generated by a realistic liquid simulator, and so we expect that this will carry over to the data we collected for these experiments. We trained the recurrent LSTM CNN using the mini-batch gradient descent method Adam [26] with a learning rate of 0.0001 and default momentum values, for 61,000 iterations. We unrolled the recurrent network during training for 32 frames and used a batch size of 5. We scaled the input images to 400×300 resolution. The error signal was computed using softmax loss. As in [7], we found the best results are achieved when we first pre-train the network on crops of liquid in the images, and then train on full images.

Figure 5b shows the performance of the detection network, and the image in Figure 2d shows an example of the output of the network. These results clearly show that our detection network is able to classify pixels with high precision and recall. This suggests that the network will work well for

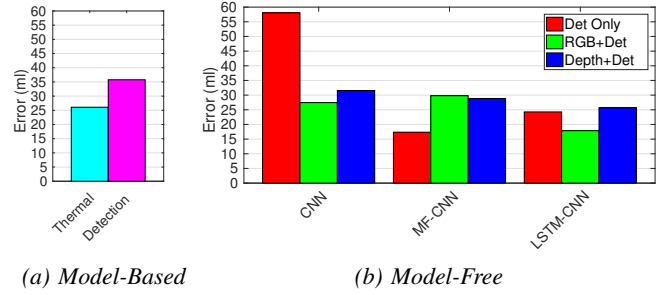


Fig. 6: Root mean squared error in milliliters of each of the methods for volume estimation. The left plot shows the error for the model-based method using either the thermal image or the output of the detection network as pixel labels. The right plot shows the error for the model-free methods when the networks take as input only the liquid detections (red), the liquid detections plus the color image (green), and the liquid detections plus the depth image (blue).

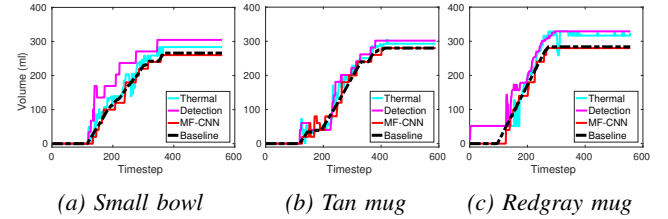


Fig. 7: The volume estimates for the two model-based methods and the multi-frame detection only model-free network. The black dashed line is the baseline ground truth. We randomly selected one sequence for each target container from our test set to display here. Best viewed in color.

estimating the volume of liquid in the target container. We should note, however, that due to the relatively small size of the training set, this detection network will work well only for the tasks described in this paper and will not generalize to other environments or tasks.

E. Comparing Methods for Volume Estimation

For our model-free methodology, every network was trained using the mini-batch gradient descent method Adam [26] with a learning rate of 0.0001 and default momentum values. Each network was trained for 61,000 iterations, at which point performance tended to plateau. All single-frame networks were trained using a batch size of 32; all multi-frame networks with a window of 32 and batch size of 5; and all LSTM networks with a batch size of 5 and unrolled for 32 frames during training. The input to each network was a 160×160 resolution crop of either the liquid detections only, the color image and detections appended channel-wise, or the depth image and detections appended channel-wise. We discretize the output to 100 values for the range of 0 to 400ml (none of our experiments use volumes greater than 400ml) and train the network to classify the volume. The error signal was computed using the softmax with loss layer built into Caffe [21]. In our data we noticed that approximately $\frac{2}{3}$ of the time during each pouring sequence was spent either before or after pouring had occurred, with little change in the volume. We found that the best results could be achieved by first pre-training each network on data from the middle of each sequence during which the volume was actively changing, and then training on data sampled from the entire sequence.

We discretize V_t and the output of the network into 20 values³ and compute $P(z_t|v_t)$ used in the model-free method for all z_t and v_t from the output of the networks on the data (we compute a separate observation probability distribution for each network).

Figure 6 shows the root mean squared error in milliliters on the testing data for each method with respect to our baseline ground truth comparison described in section III-E.3. It should be noted that although both our baseline ground truth estimate and the thermal estimate in Figure 6a are derived from the same data, the difference between the two can be largely attributed to the fact that the baseline method is able to look backwards in time and adjust its estimates, whereas the thermal model-based method can only look forward (which is necessary for control). For example, in the initial frames of a pour, as the water leaves the source container, it can splash against the side of the target container, causing the forward thermal estimate to incorrectly estimate a spike in the volume of liquid, whereas the baseline method can smooth this spike by propagating backwards in time.

While the error for both model-based methods are relatively small, it is clear that many of the model-free methods are actually better able to estimate the volume of liquid in the target container. Surprisingly, the best performing model-free estimation network is the multi-frame network that takes as input only the pixel-wise liquid detections from the detection network. The networks trained on detections only are the only networks that receive no shape information about the target container (both the depth and color images contain some information about shape), so intuitively, it would be expected that they would be unable to estimate the volume of more than a single container, and thus perform more poorly than the other networks. However, a lot of the temporal and perceptual information used by our methodology is already provided in the pixel-wise liquid detections, thus temporal information in addition to either color or depth images are not as beneficial to the networks.

We can verify that this is indeed the case by looking at the volume estimates on randomly selected pouring sequences from the test set, one for each target container. Figure 7 shows the volume estimates for the two model-based methods and the multi-frame detection only method as compared to the baseline. It is clear from the plots that the multi-frame network is better able to match the baseline ground truth than either of the model-based methods. Not only does the multi-frame network outperform the model-based methods, but unlike them, it does not require either an expensive thermal camera or a model of the target container. For these reasons, we utilize this method in the next section for carrying out actual pouring experiments with closed-loop visual feedback.

F. Pouring with Raw Visual Feedback

Estimating the volume *a posteriori* and using a volume estimator as input to a pouring controller are two very

³While this may seem rather coarse, we found it works well in practice.

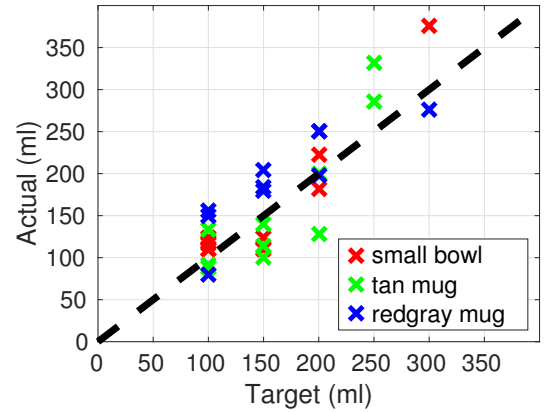


Fig. 8: Plot of the result of each pour using our model-free method as input to the controller. The x-axis is the target amount that the robot was attempting to reach, and the y-axis is the actual amount the robot poured. The points are color-coded by the target container. The black dashed line shows a 1:1 correspondence for reference.

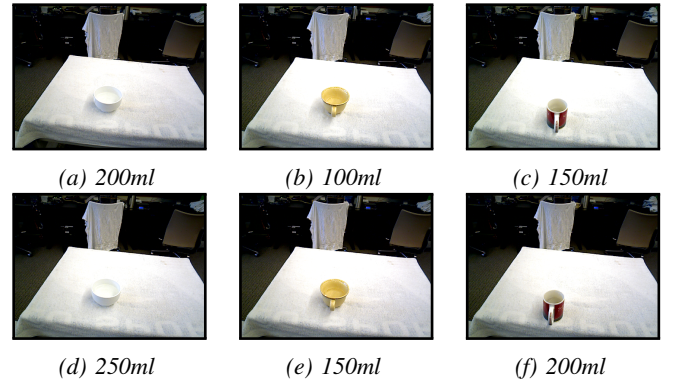


Fig. 9: Reference images for each of the three target containers. This is exactly the perspective the robot sees when looking at the containers. Notice that a 50ml difference is difficult to perceive even for a human (best viewed enlarged).

different problems. A volume estimation method may work well analyzing the data after the pouring is finished, but that does not necessarily mean it is suitable for control. For example, if the estimator outputs an erroneous value at one timestep, it may be able to correct in the next since the trajectory of the pour does not change. However, if this happens during a pour and the estimator outputs an erroneous value, this may result in a negative feedback loop in which the trajectory deviates more and more from optimal, leading to more erroneous volume estimates, etc. To verify that our chosen method from the previous section is actually suitable for control, we need to execute it on a real robot for real-time control.

To test the multi-frame network with detections only, we executed 30 pours on the real robot using the PID controller described in section III-D. We ran 10 sequences on each of the three target containers. For each sequence, we randomly selected a target volume in $\{100, 150, 200, 250, 300\}$ milliliters and we randomly initialized the volume of water in the source container as either 300, 350, or 400 milliliters, always ensuring at least a 100ml difference between the starting amount in the source and the target amount (so the robot cannot simply dump out the entire source and call it

a success). Each pour lasted exactly 25 seconds, and we evaluated the robot based on the actual amount of liquid in the target container (as measured by a scale) after the pour was finished.

Figure 8 shows a plot of each pour, where the x-axis is the target amount and the y-axis is the actual volume of liquid in the target container after the pour finished. Note that the robot performs approximately the same on all containers. This is particularly interesting since the volume estimation network is never given any information about the target container, and must simply infer it based on the motion of the liquid. Additionally, almost all of the 30 pours were within 50ml of the target. In fact, the average error over all the pours was 38ml. For reference, Figure 9 shows 50ml differences for each of our 3 containers from the robot’s perspective. As is apparent from this figure, 50ml is a small amount, and a human solving the same task would be expected to have a similar error.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduce a framework for visual closed-loop control for pouring specific amounts of liquid into a container. To provide real-time estimation of the amount of liquid poured so far, we develop a deep network structure that first detects the presence of water in individual pixels of color videos and then estimates the volume based on these detections. To generate the data and labels required to train the deep networks, we collect training videos with heated water observed by both an RGB-D camera and a calibrated thermal camera. A model-based approach allows us then to estimate the volume of liquid in a container based on the pixel-level water detections.

Our experiments indicate that the deep network architecture can be trained to provide real-time estimates from color only data that are slightly better than the model-based estimates using thermal imagery. Furthermore, once trained on multiple containers, our volume estimator does not require a matched shape model of the target container any more. We incorporated our approach into a PID controller and found that it on average only missed the target amount by 38ml. While this is not accurate enough for some applications (e.g., some industrial settings), it is well suited for similar pouring tasks in standard home environments and is on par with what a human would be expected to do in a similar setting. To our knowledge, this is the first work that has combined visual feedback with control in order to pour specific amounts of liquids into everyday containers.

This work opens up various directions for future research. One important avenue is to develop methods to improve the robustness of the neural networks. Currently, because of the limited size of our dataset, the networks only work properly on the three target containers they were trained on. We believe that the networks will be able to generalize to arbitrary containers when trained on sufficiently many examples, but this still has to be shown. Other interesting directions include generalization to different liquids, such as pouring a glass of soda or a cup of coffee, representing target amounts in *relative* terms, such as in “Pour me a half cup

of water”, and more sophisticated control schemes on top of our perception.

REFERENCES

- [1] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, “Learning to place new objects in a scene,” *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1021–1043, 2012.
- [2] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, “A geometric approach to robotic laundry folding,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.
- [3] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. Allen, “Multi-sensor surface analysis for robotic ironing,” *arXiv preprint arXiv:1602.04918*, 2016.
- [4] Z. Xu and M. Cakmak, “Enhanced robotic cleaning with a low-cost tool attachment,” in *IROS*, 2014.
- [5] A. Yamaguchi and C. G. Atkeson, “Differential dynamic programming with temporally decomposed dynamics,” in *Humanoids*, 2015, pp. 696–703.
- [6] A. Yamaguchi and C. Atkeson, “Stereo vision of liquid and particle flow for robot pouring,” in *Research Gate (Working Paper)*.
- [7] C. Schenck and D. Fox, “Towards learning to perceive and reason about liquids,” in *ISER*, 2016.
- [8] D. Walker, L. Smarandescu, and B. Wansink, “Half full or empty: Cues that lead wine drinkers to unintentionally overpour,” *Substance Use & Misuse*, vol. 49, no. 3, 2014.
- [9] L. Kunze and M. Beetz, “Envisioning the qualitative effects of robot manipulation actions using simulation-based projections,” *Artificial Intelligence*, 2015.
- [10] J. Langsfeld, K. Kaipa, R. Gentili, J. Reggia, and S. Gupta, “Incorporating failure-to-success transitions in imitation learning for a dynamic pouring task,” in *IROS Workshop on Compliant Manipulation*, 2014.
- [11] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba, “Vision based behavior verification system of humanoid robot for daily environment tasks,” in *Humanoids*, 2006, pp. 7–12.
- [12] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, “Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives,” *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011.
- [13] M. Cakmak and A. L. Thomaz, “Designing robot learners that ask good questions,” in *HRI. ACM*, 2012, pp. 17–24.
- [14] L. Roza, P. Jimenez, and C. Torras, “Force-based robot learning of pouring skills using parametric hidden markov models,” in *RoMoCo*, 2013, pp. 227–232.
- [15] C. Schenck and D. Fox, “Guided policy search with delayed sensor measurements,” *arXiv preprint arXiv:1609.03076*, 2016.
- [16] S. Griffith, V. Sukhoy, T. Wegter, and A. Stoytchev, “Object categorization in the sink: Learning behavior-grounded object categories with water,” in *Proceedings of the 2012 ICRA Workshop on Semantic Perception, Mapping and Exploration*. Citeseer, 2012.
- [17] A. Rankin and L. Matthies, “Daytime water detection based on color variation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 215–221.
- [18] A. L. Rankin, L. H. Matthies, and P. Bellutta, “Daytime water detection based on sky reflections,” in *ICRA*, 2011.
- [19] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015, pp. 3431–3440.
- [20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [22] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” *arXiv preprint arXiv:1503.04069*, 2015.
- [23] R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *ICRA*, 2011.
- [24] P. Pinggera, T. Breckon, and H. Bischof, “On cross-spectral stereo matching using dense gradient features,” in *CVPR*, 2012.
- [25] P. Blunsom, “Hidden markov models,” *Lecture notes, August*, vol. 15, pp. 18–19, 2004.
- [26] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.